DM6812/DM6912 Driver for Windows 98/NT/2000/XP

User's Manual

version 4.1

RTD Embedded Technologies, Inc.
"Accessing the Analog World"®

SWM-640010006
Rev. B

ISO9001 and AS9100 Certified

RTD Embedded Technologies, INC.
103 Innovation Blvd.
State College, PA 16803-0906

Phone: +1-814-234-8087
FAX: +1-814-234-5218

E-mail
sales@rtd.com
techsupport@rtd.com

web site
http://www.rtd.com

Revision History

08/06/2003	revision A issued
09/24/2003	Revision B issued Version 4.1 released Company name changed Windows 95 support removed All function and files names *6812 changed to *6812

DM6812/DM6912 Driver for Windows 98/NT/2000/XP
Published by:

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College, PA 16803-0906

Copyright 2003 RTD Embedded Technologies, Inc.
All rights reserved
Printed in U.S.A.

The RTD Logo is a registered trademark of RTD Embedded Technologies. cpuModule and utilityModule are trademarks of RTD Embedded Technologies. PS/2, PC/XT, PC/AT and IBM are trademarks of International Business Machines Inc. MS-DOS, Windows, Windows 98, Windows NT, Windows 2000 and Windows XP are trademarks of Microsoft Corp. PC/104 is a registered trademark of PC/104 Consortium. All other trademarks appearing in this document are the property of their respective owners.

Table of Contents

TABLE OF CONTENTS **4**

INTRODUCTION **5**

INSTALLATION **6**

INSTALLATION OF THE DRIVER AND EXAMPLE PROGRAMS 6
THE DM6812 BOARD DRIVER 7
THE DRIVER CONFIGURATION 7
INSTALLING A NEXT DM6812 BOARD IN THE SYSTEM 8
WORKING WITH OTHER BOARDS USING THE WINRT DRIVER 8
USING WINDOWS REGISTRY KEYS 9

THE DRIVER API FUNCTIONS **11**

USING THE DRIVER WITHOUT THE WINDOWS-GUI **12**

DRIVER API FUNCTION GROUPS **13**

DRIVER INITIALIZATION FUNCTIONS 13
GENERAL BOARD CONTROL FUNCTIONS 13
DIGITAL I/O FUNCTIONS 13
INTERRUPT HANDLING 13
USER TIMER-COUNTER FUNCTIONS 13
ERROR HANDLING FUNCTIONS 13

ALPHABETICAL DRIVER API FUNCTIONS REFERENCE **14**

EXAMPLE PROGRAMS REFERENCE **26**

WIN32 WINDOWS APPLICATIONS 26

LIMITED WARRANTY **27**

Introduction

The DM6812 Windows 98/NT4/2000/XP driver was designed for programmers who write Windows-based application programs with using the RTD's DM6812 board or DM6912 board. DM6812 and DM6912 are the same boards with different interface connector. Later we will refer both boards as DM6812.

The driver provides an Application Programming Interface with a lot of function calls to perform all the data acquisition tasks of the board users.

The board driver based on the BlueWater System's WinRT device driver kit.

There are example programs to demonstrate the various board features and the usage of the driver API. The example programs are written in Microsoft Visual C++ ver. 6.0.

Installation

Installation of the Driver and Example Programs

Before installing the driver and example program files, you need to install the DM6812 board in your PC. Please follow the instructions of the manufacturer, how to install the board in a computer.

To install the drivers for the DM6812 under Windows 98/2000/XP you need the installation diskette 0. The readme.txt file on this diskette describes the necessary installation steps.

For NT4 the installation diskette 0 is not needed.

On the installation diskette 1 you can found the **Setup.exe** program, which installs on your PC the DM6812 board driver and example programs. The setup program automatically detects your operating system and installs the appropriate files on your PC.
After starting the setup, please follow the instructions on the screen to install the programs. You can select the directory where to install the files.
The setup also adds to the 'Start menu' under the 'Programs' folder of your Windows system the 'RTD Embedded Technologies' folder. It contains shortcuts to the example programs, the driver configuration utility and the readme.txt file.

The example programs will be installed in the target directory under the 'Examples' folder. This folder contains the example program sources for the DM6812 board, and the project files for Microsoft Visual C++ users to rebuild the programs. The example programs are compiled with the Microsoft Visual C++ ver. 6.0. In case of different version of Visual C is installed on your PC, please rebuild the executable files.

Uninstallation: you can uninstall the DM6812 driver and example programs under the 'Control panel' with the 'Add/Remove Programs' tool.

The DM6812 board driver

The RTD's DM6812 board driver handles the hardware through the BlueWater's WinRT driver. The WinRT driver provides the low-level access to the board, and the RTD's drvr6812.dll provides the device API for the programmers, and communicates with the hardware through the WinRT driver.

The RTD board drivers has multi-board feature. It means that it is possible to use more than one board in the system.

The base address of the DM6812 board can be set by DIP-switch. If you are using multiple boards in your system, you need assign a different base address to each board. The IRQ channel will be set programmatically. When you install the driver for the board, the installation program sets up a default IRQ channel, and during the operating of the board must be used this IRQ number.
Each board requires a unique WinRT device to handle the hardware. The maximum number of devices is 32.

The installation program creates the wrtdev.txt file in your 'Examples' directory. It contains the WinRT device number used with the installed board. The example programs are starting with reading of this file to obtain the proper device number.

During the installation the Setup.exe program set the default driver values:
Board 1: 0x300 base address, IRQ 0x11

If these settings are not proper to your PC configuration, you can modify the settings with the **Drvr_cfg.exe** program.

The Driver Configuration

As in the previous chapter mentioned, after installing the driver the default settings are used.

If these settings are not proper to your PC configuration, or there are other RTD boards used, you can modify the settings.
Under Windows NT driver can be configured with the **Drvr_cfg.exe** program. This program is installed on your PC during the setup process and it can be started from the Start menu.

Program options:

- Device number: The number of device, which handles the board. Each board has a unique device number between and 0-31.
- I/O port base address: The I/O base address of the board.
- IRQ number: The IRQ number using of the board.

- Register: Press this button to store the settings in the Windows registry.
- Refresh: Press this button to get the information from the registry about the selected device number.

Under Windows 98/2000/XP same changes can be made using Control Panel/System/[Hardware]/Device Manager/WinRT Devices.

Do not forget, that all boards are required different driver devices, and in your C++ program the DEVICE_NO must be used according to the driver device number.

Installing a next DM6812 board in the system

It is possible to use more than one board in the system using the RTD's data acquisition board drivers. Please follow the next steps to install a multi-board – system:

- Install the first board with all software and make sure that everything is working well.
- Place the second board in an empty slot in the PC, and switch on the computer.
  Windows NT4: After the system startup start the installation program and answer 'YES' to the next board installation question. Restarting the system with the two-board example program is possible to test the working of both boards. Please note that during the second board installation the driver and example program files will be refreshed thus you need to save the important changes before starting the install.
- *Windows 98/ME/2000/XP:* You need only the installation diskette 0 to install the next boards. Follow the steps described in the readme.txt file on diskette 0.

Working with Other Boards Using the WinRT Driver

As you may have in your computer other RTD boards, which uses the WinRT driver, or a board from other manufacturer with WinRT, our installation program automatically detects the WinRT settings from the system registry file, and installs the appropriate (next) WinRT device. You can see an informational dialog box about this issue during the setup.
As an application written for the DM6812 board should know which WinRT device is assigned to the board, the installation program generates the file wrtdev.txt in the 'Examples' directory containing the actual device number. The example programs are reading this file at the startup to identify this number.
If the other board in your system, which uses the WinRT driver does not handle the situation when there is other WinRT-based application, install this board firstly, and then install the DM6812 which handles correctly this state.

Using Windows registry keys

The RTD board drivers are using the next registry keys:

Windows NT 4.0:

```
HKEY_LOCAL_MACHINE
|
| -- System
     |
     | -- CurrentControlSet
          |
          | -- Services
               |
               | -- WinRT
                    |
                    | -- WinRTdev0       (one per device)
                    |     |
                    |     | -- Parameters
                    |           |
                    |           | -- Section0

                    | -- WinRTdev1
                    |     |
                    |     | -- Parameters
                    |           |
                    |           | -- Section0
                    |
```

Windows 98 Enum key path:

```
HKEY_LOCAL_MACHINE
|
| -- Enum
```

Windows 98 Class key path:

```
HKEY_LOCAL_MACHINE
|
| -- System
     |
     | -- CurrentControlSet
          |
          | -- Services
               |
               | -- Class
                    |
                    | -- WinRTDevices
```

Windows 2000/XP device class information:

```
HKEY_LOCAL_MACHINE
|
| -- System
     |
     | -- CurrentControlSet
          |
          | -- Control
               |
               | -- Class
                    |
                    | -- {D695ED6A-630D-4D83-D8B-F1F0AC107AD0}
                    |
                    | -- 0001
```

Windows 2000/XP Plug and Play devices are enumerated under the Enum key:

```
HKEY_LOCAL_MACHINE
|
| -- System
   |
   | -- CurrentControlSet
        |
        | -- Enum
             |
             | -- PCI
```

The Driver API Functions

The resources on the DM6812 board can be accessed from Windows through the driver API (Application Programming Interface) functions. The executable code of these functions is located in the drvr6812.dll file. To write applications using the API functions you must include the reg6812.h header file, and link the program with the drvr6812.lib import library file.
In the example programs you can find different examples how to use the driver.

Using the Driver without the Windows-GUI

This driver is based on the Win32 system, and can run only under Windows. However, the user, who is not familiar with the Windows Graphical User Interface, can use the driver API functions in Win32 console applications, which has an MS-DOS like text-mode interface.

In console applications is not necessary to use the Windows graphical environment, but all the driver's API functions are accessible.
The Microsoft Visual C++ compiler supports writing console applications.

Driver API Function Groups

Driver Initialization Functions

OpenBoard6812
CloseBoard6812

General Board Control Functions

InitBoard6812

Digital I/O Functions

ReadDIO6812
WriteDIO6812
SelectDIOClock6812
LoadDIOBitDir6812
LoadDIOPortDir6812
LoadDIOComp6812
LoadDIOMask6812

Interrupt Handling

ClearBoardIRQ6812
ClearDIOChip6812
ClearDIOIrq6812
EnableBoardIRQ6812
IRQSharing6812
InstallCallbackIRQHandler6812
InstallCounterIRQHandler6812
RemoveIRQHandler6812
GetIRQCounter6812
IrqStatus6812
SetIRQPolarity6812
EnableDIOIrq6812
SetDIOIrqMode6812

User Timer-Counter Functions

ClockDivisor6812
ClockMode6812
DoneTimer6812
InitTimer6812
isClockDone6812
ReadClock6812

Error Handling Functions

GetErrorStatus6812

Alphabetical Driver API Functions Reference

C

ClearBoardIRQ6812

Description:

This routine is used to clear the P14 IRQ circuitry.

Parameters:

hBoard:	device handle

ClearDIOChip6812

Description:

This routine clears the selected DIO chip.

Parameters:

hBoard:	device handle
Chip:	0, 1, 2 (selects which DIO chip on board)

ClearDIOIRQ6812

Description:

This routine clears the selected DIO chips IRQ status bit.

Parameters:

hBoard:	device handle
AIOChip:	0, 1, 2 (selects which DIO chip on board)

ClockDivisor6812

Description:

The ClockDivisor procedure is used to set the divisor of a designated counter on the 8254 programmable interval timer (PIT). This procedure assumes that the counter has already been set to receive the least significant byte (LSB) of the divisor followed by the most significant byte (MSB).

Parameters:

hBoard:	device handle
ATimer:	0,1,2
ADivisor:	divisor

ClockMode6812

Description:

The ClockMode procedure is used to set the mode of a designated counter on the 8254 programmable interval timer (PIT).

Parameters:

hBoard:	device handle
ATimer:	0,1,2
AMode:	0,1,2,3,4,5

CloseBoard6812

Description:

This routine is used to close board driver.

Parameters:

DeviceNumber:	WinRT device number to load
szBuf:	message buffer

Return value:

TRUE on success, FALSE on error

DoneTimer6812

Description:

Shut down timer operation initiated by 'InitTimer'.

Parameters:

hBoard: device handle

EnableBoardIRQ6812

Description:

This routine enables/disables the board P14 IRQ.

Parameters:

hBoard:	device handle
AEnable:	0 = disable
	1 = enable

EnableDIOIRQ6812

Description:

This routine enables the selected DIO chips interrupt.

Parameters:

hBoard:	device handle
AIOChip:	0, 1, 2 (selects which DIO chip on board)
AEnable:	0 = Disabled
	1 = Enabled

GetErrorStatus6812

Description:

This routine is used to get the error code and string in case of errors.
The function clears the last error code.

Parameters:

hBoard:	device handle
ErrorCode:	Error Code
ErrorString:	Return error string

Return Value:

TRUE	if error occurred
FALSE	otherwise

GetIRQCounter6812

Description:

This routine is used to get the counter incremented by IRQ.

Parameters:

hBoard:	device handle
irqsetup:	return parameters to caller

InitBoard6812

Description:

This routine should always be called first.
This clears the board and variables the driver uses.

Parameters:

hBoard:	device handle

InitTimer6812

Description:

The InitTimer function is used to set the user clock to generate the designated output rate. The order of programming the timers are important to ensure loading the timers fast enough. The maximum attainable rate this function, as written, is 1000 Hz although you can easily change this by adjusting the divisors accordingly.

Parameters:

hBoard:	device handle
ARate:	rate

Return value:

TRUE on success, FALSE on error

InstallCallbackIRQHandler6812

Description:

This routine is used to install IRQ handler function.

Parameters:

hBoard:	device handle
isrT handler:	user callback function

Return value:

TRUE on success, FALSE on error

InstallCounterIRQHandler6812

This routine is used to install IRQ handler function.

Parameters:

hBoard:	device handle

irqsetup: return parameters to caller

Return value:

TRUE on success, FALSE on error

IrqSharing6812

Description:

This routine enables and disables the interrupt sharing feature of the board.

Parameters:

hBoard: device handle
AEnable: 0 = Disabled
1 = Enabled

IrqStatus6812

Description:

This routine returns the IRQ status from the board.

Parameters:

hBoard: device handle

Returns:

the status register value

---

isClockDone6812

Description:

The isClockDone function returns TRUE if the designated counter on the 8254 programmable interval timer (PIT) has overflowed. The function uses the "counter latch command" to store the 16 bit counter register for the separated reading (low byte first, high byte next). The function stores the previous state of the counter register in the 'PrevClock' global variable, and returns TRUE, when the current counter register is larger than the previous one, i.e. the counter is loaded back. This ensures the recognition of the counter owerflow even of high clock rates.

Parameters:

hBoard: device handle

Returns:

TRUE / FALSE

LoadDIOBitDir6812

Description:

Load a Bit programable IOport. Ports 0,2,4 are Bit programable on the DM6812.

Parameters:

hBoard:	device handle
AIOPort:	port 0-6
Adir0-Adir7:	direction 0 = input 1 = output

LoadDIOComp6812

Description:

Load DIO compare register.

Parameters:

hBoard:	device handle
AIOChip:	chip 0-2
AData:	compare value

LoadDIOMask6812

Description:

Load DIO mask register.

Parameters:

hBoard:	device handle
AIOChip:	chip 0-2
AData:	mask bits

LoadDIOPortDir6812

Description:

Load a Port programable IOport. Ports 1,3,5 are Port programable on the DM6812.

Parameters:

hBoard:	device handle
AIOPort:	port 0-6
ADir:	direction 0 = input 1 = output

Description:

This routine is used to open board driver.

Parameters:

DeviceNumber:	WinRT device number to load
LoadDevice:	Indicate to load WinRT driver or not
szBuf:	Message buffer
BoardConfig:	Return board configuration

Return value:

TRUE on success, FALSE on error

ReadClock6812

Description:

Read the timer/counter count value.

Parameters:

hBoard:	device handle
ATimer:	number of the Timer in the range of 0..2.

Returns:

16-bit count value.

ReadDIO6812

Description:

The ReadDIO function returns the value of the specified digital input port. Each digital input line is represented by a bit of the return value. Digital in 0 is bit 0, digital in 1 is bit 1, and so on.

Parameters:

hBoard:	device handle
AIOPort:	port 0 – 6

Returns:

8 bit unsigned character value

RemoveIRQHandler6812

Description:

This routine is used to uninstall IRQ handler function.

Parameters:

hBoard:	device handle

SetIRQPolarity6812

Description:

This routine sets the P14 interrupt to use the positive edge or negative edge.

Parameters:

hBoard:	device handle
Polarity:	0 = positive edge
	1 = negative edge

SelectDIOClock6812

Description:

Select digital sample clock.

Parameters:

hBoard:	device handle
AIOChip:	0, 1, 2 (selects which DIO chip on board)
Clock:	0 = 8 MHz System
	1 = Programmable clock

SetDIOIrqMode6812

Description:

Set digital IRQ mode.

Parameters:

hBoard:	device handle
AIOChip:	chip 0 – 2
Amode:	IRQ mode 0=EVENT 1=MATCH

WriteDIO6812

Description:

The WriteDIO function writes 'Data' to the specified digital output port.

Parameters:

hBoard:	device handle
AIOPort:	port 0 - 6
Data:	0 - 255

Example Programs Reference

Win32 Windows applications

Name	Feature	Remarks
Digital	Sample program that demonstrates how to program digital I/O.	Sample program that demonstrates how to read and write the digital I/O. The program sets Ports 0 and 1 as outputs. The rest of the ports are set as inputs. The program constantly displays the value of the lines on the input port in the center of the screen. You can enter a value to be sent to the output port. Wiring the input port lines to the corresponding output port lines will allow you to change the output lines and view the changes at the same time.
Digital_IT	Sample program that demonstrates how to use the P0.0 input as an interrupt source and how to write an Interrupt Service Routine (ISR).	Sample program that demonstrates how to read and write the digital I/O using interrupts. The program sets Port 0 as input. The rest of the ports are set as outputs. This program sets up the P0.0 input to generate interrupts. An interrupt service routine (ISR) is provided that simply sends a value out the output port P2. The values are displayed on the screen as a foreground process. Although the ISR in this program is trivial, all the ingredients for powerful interrupt driven data acquisition are included. The timer is programmed to run at the specified rate. If you connect the output from timer/counter 2 to P0.0, interrupts will be generated at the timer rate.
Interrupt	Sample program that demonstrates how to use the timer as an interrupt source and how to write an Interrupt Service Routine (ISR).	Sample program that demonstrates how to use the Timer/Counter to generate interrupts. The program sets Port 0 as output. The rest of the ports are set as inputs. An interrupt service routine (ISR) is provided that simply sends an incremented counter value out the output port P0. The values are displayed on the screen as a foreground process. Although the ISR in this program is trivial, all the ingredients for powerful interrupt driven data acquisition are included. The timer is programmed to run at the specified rate.
Match	Sample program that demonstrates how to use the P0.0 input as match mode interrupt source.	Sample program that demonstrates how to read and write the digital I/O using match mode interrupts. The program sets Port 0 as input. The rest of the ports are set as outputs. This program sets up the P0 input port to generate interrupts when the input at the port match with the compare register value. An interrupt service routine (ISR) is provided that simply sends an incremented value out the output port P2. A Windows timer is programmed to run at the specified rate. It increments the compare register valu. If you connect the outputs of P2 to the inputs of P0, interrupts will be generated.
Timers	Sample program that demonstrates how to program the user timer.	This program will illustrate how to program the 8254 timer on the DM6812. Timer 0 and 1 are programmed to generate a 10 Hz pulse train. Timer 2 is used to count these pulses. The result is converted into Seconds and printed on the screen. Make sure the jumpers for connecting the output from Timer 0 to the input of Timer 1 and the output from Timer 1 to the input of Timer 2 are in place.

Limited Warranty

RTD Embedded Technologies, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from RTD Embedded Technologies, INC. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, RTD Embedded Technologies will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to RTD Embedded Technologies. All replaced parts and products become the property of RTD Embedded Technologies. Before returning any product for repair, customers are required to contact the factory for an RMA number.

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by RTD Embedded Technologies, "acts of God" or other contingencies beyond the control of RTD Embedded Technologies), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN RTD Embedded Technologies. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND RTD Embedded Technologies EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MECHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO THE DURATION OF THIS WARRANTY. IN THE EVENT THE PRODUCT IS NOT FREE FROM DEFECTS AS WARRANTED ABOVE, THE PURCHASER'S SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. UNDER NO CIRCUMSTANCES WILL RTD Embedded Technologies BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, AND SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College PA 16803-0906
USA
Our website: www.rtd.com